(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 776 107 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
28.05.1997 Bulletin 1997/22

(51) Int Cl.[6]: **H04L 12/18**, H04L 29/06

(21) Application number: **96308040.3**

(22) Date of filing: **06.11.1996**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **06.11.1995 US 7262**
16.01.1996 US 586136

(71) Applicant: **XEROX CORPORATION**
Rochester New York 14644 (US)

(72) Inventors:
- **Curtis, Pavel**
  Los Altos CA 94022 (US)
- **Nichols, David A.**
  Mountain View CA 94043 (US)
- **Dixon, Michael D.**
  San Francisco CA 94110-4317 (US)
- **Frederick, Ronald A.**
  Mountain View CA 94040 (US)

(74) Representative: **Phillips, Margaret Dawn et al**
Rank Xerox Ltd
Patent Department
Parkway
Marlow Buckinghamshire SL7 1YL (GB)

(54) Multimedia coordination system

(57) In a network, a media coordination system (2) provides secure multimedia communication channels in a collaborative network environment. The media coordination system provides automatic encryption, dynamic interconnection of streams of data, and user interface elements that provide users with control over the ultimate destination of their audio and video data. The infrastructure of the system (2) includes a plurality of client workstations (4) that are connected to a central server (22) using point-to-point network connections. The central server (22) maintains a persistent virtual world of network places with objects located therein. Streams of audio and video data are coordinated between client workstations (4) operating in the persistent virtual world by a key manager object using channels, transmitters, and receivers. The client workstations (4) multicast their audio and video data over the network to defined recipients after receiving a multicast address and an encryption key for a specific multicast channel. In order to protect the privacy of all communications and the integrity of the coordination system (2), each client workstation (4) retains significant control over distribution and reception of audio and video data since multicast transmission is tied to specific user interface elements. The multimedia user interface elements include cameras (18), speakers (16), microphones (20), and video panes (12). Since the central server (22) only coordinates where audio and video data is broadcast for a particular interface element, each client workstation (4) ultimately controls the destination of multimedia data through selection of the element at the user interface.

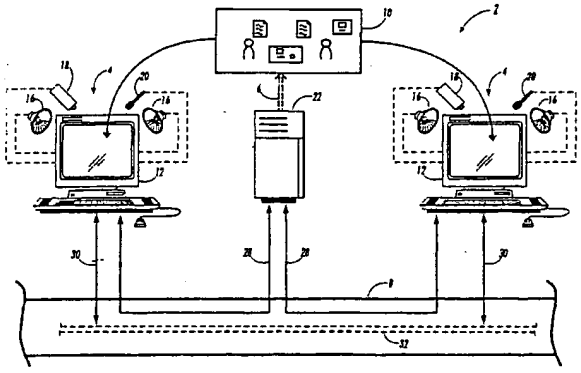

FIG. 1

EP 0 776 107 A2

whether each user can be interrupted. In effect, communication using audio and video data is advantageously used in the collaborative environment to increase productivity between network users in the collaborative environment.

In accordance with one aspect of the invention, there is provided a method for dynamically controlling multiple channels of data in a multi-user collaborative system having a central server connected to a plurality of client workstations over a network. The method includes the steps of: displaying at each client workstation a view on a room object stored in an object database on the central server, the room object being associated with a first channel stored in the object database; providing, at each client workstation, visual identification of each user object located in a virtual room, each pair of user objects located in the virtual room having associated therewith a whisper channel; initiating, at a first client workstation, broadcast of data to each user object located in the virtual room by selecting a first interface element displayed at the first client workstation, the first interface element being associated with the room object and directing data to the first channel; and interrupting, at the first client workstation, broadcast of data transmitted over the first channel by selecting a second interface element displayed at the first client workstation, the second interface element being associated with a user object at a second client workstation, the interrupting step initiating broadcast of data at the first client workstation to the whisper channel associated with the user object at the second client workstation.

In accordance with a second aspect of the present invention, there is provided a system for coordinating communication of data between each of a plurality of client workstations adapted to broadcasting data, in a network interconnecting a central server with a memory and the plurality of client workstations, the system comprising: a device for receiving data at a first client workstation; a first transmitter for coordinating transmission of data from said device over a channel, said first transmitter being stored in the memory of the central server; a first receiver for coordinating receipt of data over the channel at a second client workstation, said first receiver being stored in the memory of the central server; means for providing a first encryption key to the first client workstation and the second client workstation for secure broadcast of data over the channel; means for providing a second encryption key to the first client workstation and the second client workstation in response to a third client workstation storing in the memory of the central server a second receiver for coordinating receipt of data over the channel at the third client workstation, said providing means ensuring secure broadcast of data over the channel to the first client workstation, the second client workstation, and the third client workstation.

In another aspect of the invention, there is provided in a network interconnecting a central server and a plurality of client workstations adapted to sending and receiving data, a method for coordinating communication of data between each of the plurality of client workstations. The method includes the steps of associating a first client workstation with a device, the device providing multimedia input at the first client workstation; defining a first transmitter in a memory of the central server for transmitting data from the device over a first channel; defining a first receiver in the memory of the central server for receiving audio signals over the first channel at a second client workstation; providing a first encryption key to the first client workstation and the second client workstation to provide secure communication of data over the first channel; defining, subsequent to the providing step, a second receiver in the memory of the server for receiving audio signals over the first channel at a third client workstation; and altering, in response to the defining step, the first encryption key provided to the first client workstation and the second client workstation, the altering step providing a second encryption key to the first client workstation, the second client workstation, and the third client workstation for communication of data over the first channel so that communication broadcast over the first channel is secure.

In yet another aspect of the invention, there is provided a method of coordinating multicast audio data between a plurality of client workstations connected over a network, each client workstation having a point to point connection with a central server. The method includes the steps of displaying a communicator at a client workstation, the communicator providing a first user interface element to direct audio data from an audio device at the client workstation to a first set of client workstations and a second user interface element to direct audio data from the audio device to a second set of client workstations, the second set of client workstations being a sub-set of the first set of client workstations; defining, in a memory of the central server, a public channel for transmission of audio data to the first set of client workstations and a private channel for transmission of audio data to the second set of client workstations; receiving, at the central server, a first user signal from the communicator at the client workstation to direct audio data from the audio device to the public channel; providing with the central server, in response to the first user signal, a first encryption key to the client workstation, the first encryption key enabling transmission of audio data between the client workstation and the first set of client workstations over the public channel; receiving, at the central server, a second user signal from the communicator at the client workstation to direct audio data from the audio device to the private channel; providing with the central server, in response to the second user signal, a second encryption key to the client workstation, the second encryption key enabling transmission of audio data between the client workstation and the second set of client workstations over the private channel; and toggling, at the client workstation, between the first encryption key and the second encryption key in response to a third user signal from the communicator to terminate transmission of audio data from the audio device to the private channel and the second user signal, the toggling step being performed without the client workstation communicating with the central server so that the client workstation

Each client workstation 4 is connected over physical network 8 to central server 22 using a point-to-point networking transmission control protocol (TCP), as indicated generally by connections 28. Communication between the client workstations 4 and central server 22 include typed commands from a user, control instructions from the central server to a client workstation, and status notification from a client workstation to the central server. Audio and video (A/V) data, however, is not delivered between network client workstations 4 using point-to-point network connections because A/V data is typically intended to be received by one or more network users. Instead, A/V data is efficiently "multicast" directly to network 8 by each client workstation as depicted generally by connections 30 to multicast connection or destination address 32. Client workstations 4 transmit and receive A/V data using Internet Protocol (IP) multicast and a proposed Real-time Transport Protocol (RTP). For privacy all A/V data is encrypted. IP multicast is further described by Deering et al. in "Multicast routing in datagram networks and extended LANs," ACM Transactions on Computer Systems, May 1990. RTP is disclosed by Schulzrinne et al. in "RTP: A Transport Protocol for Real-Time Applications, " IETF Internet Draft (available from ftp://ftp.internic.net/internet-drafts/draft-ietf-avt-rtp-07.txt).

A multicast routing protocol enables an individual packet to be received by many clients who have expressed an interest in the packet's specific destination address without duplicating the packet on any link. In general, a sender of such a packet is not able to control which clients are allowed to receive the packet or even discover, after the fact, which clients did receive it. By way of analogy, "multicast" transmission over network 8 is analogous to broadcasts performed using a radio transmitter. For example, each client 4 multicasts on a separate "address," which is analogous to a radio frequency, as explicitly directed by central server 22. Since the central server 22 controls the address (or frequency) of a client wanting to receive a transmission, the central server 22 is able to direct when each client is to "listen" in order to receive A/V data from other client workstations 4. Multicasting A/V data relieves the central server 22 from the task of managing large amounts of audio and video data, thereby enabling the central server 22 to manage the multicast sending and receiving addresses of client workstations 4.

Figure 2 is a detailed block diagram representation of central server 22 connected to a plurality of clients 4 (or client programs) with TCP connections 28. Central server 22 includes a programming language interpreter 24 and an object database 26. Each client 4 initiates a TCP connection 28 with server 22 which accepts and manages network connections from each client 4. The TCP connection 28 is essentially the only means of communication between client 4 and server 22. All communication on TCP connections 28 is encrypted for privacy, using the proposed Secure Sockets Layer protocol (SSL) as disclosed by Hickman in "The SSL Protocol" (available from http://home.mcom.com/newsref/std/SSL.html). In addition, the server 22 maintains the object-oriented database 26, and executes code stored in the database using interpreter 24, often in response to user commands and client protocol messages. As indicated above, the central server 22 never transmits or receives multicast data. However, central to the management of client multicast transmissions, the key manager 25, which is described in detail later, coordinates multicast data between clients 4 over multicast connections depicted generally by reference numerals 30 and 32.

Virtual objects, whether core objects 35 or world objects 34, are the basic storage unit of server 22. The server database 24 contains states and descriptions of virtual objects such as places, and tools operating in the multimedia system 2. Virtual objects are described using methods and instance variables. Examples of virtual objects include simple text documents and drawing surfaces, general tools like web browsers and tape recorders, tools designed for specific work such as data analysis, and agents that interact with other objects and users. Some of these objects are "world objects" 34 which are virtually tangible, user-visible objects like places, the things in those places, and individual users. For example, two classes of world objects include: objects modeling individual users (which are defined as "user objects") and objects modeling rooms (which are defined as "room objects"), which are indicated generally by reference numbers 36 and 37 respectively. Certain "core objects" 35 implement very general facilities that are maintained by a system administrator. For example, core objects 35 include the server side of the user interface window system 33, core room object 39, and core user object 38. Each user object 36 and room object 37 are linked to core user object 38 or core room object 39 respectively. Objects that are not core objects are objects such as room objects or user objects, or "applications" (e.g. tape recorder).

Programs written in the server's embedded programming language called "MOO" are interpreted by interpreter 24. The MOO language is disclosed by Curtis in "LambdaMOO Programmer's Manual (available as ftp://ftp.parc.xerox.com /pub/MOO/ ProgrammersManual.ps). Users invoke a MOO program each time a command is entered at server 22. In addition, the server 22 includes tools for creating new objects, and new places, and tools for modifying the behavior of objects and places. All aspects of the server database 26 are mutable during execution; objects may be created or destroyed, methods and instance variables added, modified, or removed. As a security measure, each MOO object, method, and instance variable is owned by a specified user and has access control settings. Users and the code they may own, may not, in general, modify or destroy, or in some cases inspect objects, methods, or instance variables owned by other users.

Programs running on clients 4 are primarily responsible for managing the local details of their user interface operating on display terminals 12 (shown in Figure 1). Guided mostly by commands from the server 22, a client 4 displays windows on a user's display terminal, conveys information about the user back to the server, and sends and receives

key to use solely based upon the destination address of a multicast packet. A third efficiency requirement generally mandates that the number of encryption keys a client must handle is minimized. More specifically, the third efficiency requirement mandates that the generation of encryption keys is delayed until they are actually needed. This requirement minimizes the number of cryptographic operations the server 22 must perform thereby minimizing the burden of cryptographically-secure key generation that is computationally expensive. A fourth efficiency requirement mandates that even if security considerations would otherwise permit receipt of transmission from other users, a client avoids preparing and sending A/V data unless it is actually being received by someone. In other words, there is no need to prepare A/V data for distribution to a multicast audience if no client is looking at, or listening to a particular multicast address. The fourth efficiency requirement exists in order to minimize the processing of video data, which is computationally expensive for clients to capture, compress and transmit.

A second security requirement mandates that a user at a client controls whether or not a user's audio or video signals are multicast across the network. This level of control, described in detail later, is presented to a user through appearance and behavioral attributes of an application. A third and narrower security requirement mandates that users should only risk multicasting A/V data that they explicitly authorized to be revealed to other users. For example, an application that provides A/V "tunnel" between two users enables the pair of users to communicate as they move about in different virtual places. A user should therefore only risk whatever audio or video signals that are explicitly enabled for the tunnel application to access. A user explicitly controls A/V data transmission by controlling widgets that an application includes in its graphic user interface. Four widgets, cameras, microphones, video panes, and speakers, provide control over receipt and distribution of A/V data.

C. The User Interface

Figure 3 shows a plurality of simulated window images 40 used to depict the collaborative environment 10 (shown in Figure 1). The window images 40 include console window 42, "who" window 44, "Postit" window 46, and communicator window 48. The console window 42 (or mike's console window) and communicator 48 (or mike's communicator) are two different views on the user "mike." The who window 44 provides information of all logged in users (e.g. how long they have been connected, how long they have not been active on the system, and where they are presently located). The postit window 46 is a metaphor for the Post-it ™ note. Generally, the virtual world of the collaborative environment 10 is composed of a set of interconnected places such as room objects that serve as network places to structure interaction between user objects. Each user is located in a room, and possibly the same room as other users. To a first approximation, each user sees and interacts with other users and objects located in a similar network place such as a room object. Users can control the extent of their participation in the network place by moving themselves and objects from place to place. Communication between users takes the form of typed messages, spoken audio, and live video images. Thus, users who are virtually in the same network place can see and hear each other even though there may be a considerable distance between the physical locations of the users.

The console window 42 shown in Figure 3 is one view of the user mike. Console 42 provides an overview of what "things" in this network place that are available to mike, the user. From the perspective of each user who is connected to server or "jupiter" 22, the collaborative environment 10 is a virtual world made up of rooms or locations displayed in the who window 44, as indicated generally by reference number 52. Each user is therefore in a network place or room 42. In Figure 3, user "mike" is connected to the "jupiter" server 22 and is in the "jupiter lab," as indicated on mike's console window 42. In each room or "network place," such as the jupiter lab, there may be a plurality of virtual objects and tools, such as a virtual whiteboard 54 on which pictures and diagrams are drawn, and documents such as the "jupiter notebook" 56.

Mike's console window 42 lists what objects mike is carrying, in sub-window 57, what objects are in the same room as him, in sub-window 58, and the visible exits from the room that he is in sub-window 59. Thus, user mike may select from a plurality of "exits" 59 from which he may move into another room. Sub-window 53 enables user mike to input new commands and view his history of past commands and status events. In sum, a console window 42 for any user depicts relationships between objects and users. Also, as previously noted, this collaborative environment is persistent in that these relationships between people and objects are maintained over time. The console window 42 in listing the other objects in the "jupiter lab" in sub-window 58, includes all the users who appear in the communicator window 48. Mike's communicator window 48 provides mike with another view that enables him to communicate with other users who are in the same room using audio or video data.

A communicator window 48 contains microphone widgets (such as microphone widget 63), camera widget 65, and video pane widgets (such as video pane widget 67). A speaker widget highlights when a particular user speaks. For example, video pane widget 67 and microphone widget 64 are surrounded by an invisible rectangle which represents a speaker widget 68 (shown using a dotted line). Users with video and audio turned on using buttons 63 and 65 in a particular room will appear in communicator sub-window 60. Users however that only have audio turned on or neither audio or video turned on appear in sub-windows 61 and 62 respectively. The microphone and camera widgets 63,65

a coordinator and a key generator. The key manager coordinates all A/V multicast data transmitted between each client 4 connected to server 22. Once encryption keys have been generated for a particular source of A/V data, the key manager is responsible for notifying clients 4 of appropriate multicast addresses and encryption keys for use with out-of-band communication. "Out-of-band" communication is defined herein as data that is multicast between clients, while "in-band" communication is defined herein as all data that is passed from one client to other clients through server 22.

The key manager or coordination system 25 generalizes away from A/V widgets such as microphone widget 63, camera widget 65, speaker widget 68, and video pane widget 67, since audio and video transmissions may originate from objects other than user objects. These A/V widgets are labeled as either sources or sinks, which are indicated generally by reference numbers 85 and 86 respectively. For example, camera widget 65 and microphone widget 63 of a user's database object act as a source 88, while speaker widget 68 and video pane widget 67 of a user's database object act as a sink 89. Other database objects such as a tape recorder object may act as source 90 and sink 91. The tape recorder object would act as source 90 or sink 91 depending on whether the object was being used for playback or recording.

Besides communicating with sources 85 and sinks 86, the key manager 25 communicates with channel managers 87 which can be any database object such as a room object 37 (shown in Figure 2). Sources 85, sinks 86, and channel managers 87 do not, in general, communicate with each other directly. Each object interfaces with a single coordinating object, the key manager 25. Specifically, sources 85, sinks 86, and channel managers 87, each call or are called by the key manager 25. The key manager 25 translates abstract requests made by each source, sink, or channel manager object into a multicast address and encryption keys to be used by clients 4 when sending and receiving out-of-band A/V data. In addition, the key manager coordinates the passing of any in-band data between sources 85 and sinks 86.

Figure 6 shows a more detailed block diagram of the key manager or media coordination system 25 interfacing with sources 85, sinks 86, and channel manager 87. Each source device 85 can have a plurality of devices 100 associated with it. Each device 100 is capable of generating a stream of audio or video data. For example, a user object in database 26 (shown in Figure 2) may act as source object 105. The devices associated with source 105 are devices 107 and 108 which correspond to physical audio and video inputs at a client workstation 4 (shown in Figure 2). Alternatively, a tape recorder object may act as source object 105. The tape recorder object may have one device for each stream of audio data played back. For example, source 105 would have devices 107 and 108 for two streams of audio being played back.

Sources 85 transmit to sinks 86 through channels 96. A channel is an abstraction in the object database 26 that represent pipes through which audio and video streams flow. Each channel 96 has associated with it either a static or dynamic channel membership. Each channel membership defines which sources 85 and sinks 86 have access to a particular channel. Channels 96 with a dynamic membership list have an associated channel manager 87. Each channel manager 87 is responsible for notifying the key manager 25 whenever the membership of its channel changes. In essence, the channel manager 87 decides who is allowed to be on a channel. For example, a room object acting as a channel manager 99 for channel 101 will have a channel membership list consisting of all of the objects within the room. The membership list of channel 101 changes each time a user object leaves or enters the room. For each exit and entry of the room, the channel manager 101 is required to notify the key manager 25 of each change in its membership list. Alternatively, channel 102, which does not have a channel manager, is an example of a channel with a static membership lists. Channels 96 with static membership lists are typically used when two user objects are "whispering" to each other (or carrying on a private conversation) in a room. In the case of two user objects whispering to each other, the channel membership of that channel 96 consists of just the two user objects.

1. Key Manager Interface

The key manager 25 is a central coordination system between sources 85, sinks 86, and channel managers 87. The key manager interface pertains to the manner in which the key manger 25 interacts with sources, sinks, and channel managers. Table 1 lists methods that key manager 25 provides for sources 85, and Table 2 lists notifications that key manager 25 sends back to sources 85. (Note: every method call in Table 1 passes a source object as an implicit parameter to the key manager 25. This implicit argument, however, is overridden by widget implementations; from the key manager's perspective, the calling source 85 or sink 86 always appears to be the object representing the user on whose screen a widget appears. This approach allows the key manager 25 to treat user objects in the same way as other sources 85 and sinks 86.) Each source 85 declares an intent to transmit A/V data by creating a transmitter 94 and by associating the transmitter with a device 100, using the create_transmitter() and set_transmitter_device() methods respectively. For example, source 110 declares an intent to transmit A/V data by creating transmitter 114 and by associating transmitter 114 with device 112. Generally, each source 85 creates a transmitter 94 for each reason that source may transmit A/V data. For example, a user object acting as source object 105 has transmitters 115, 116, and 117 for specific camera or microphone widgets on a user's display terminal. A transmitter is destroyed once it is no longer required using the destroy_transmitter() method.

The media coordination system is not limited to a notion of transmitters being "on" or "off" so that a new multicast address & encryption key has to be created each time a user toggles a widget button. In this on and off mode, issuing new addresses & keys add to the latency perceived by a user since a message exchange between client 4 and server 22 is required to alter where a client is broadcasting A/V data. With push-to-talk mode (as opposed to on and off mode) a client 4 is issued an individual multicast address and encryption key for sending A/V data to a single channel. Consequently, a client 4 is free to use this individual multicast address and encryption key instead of the current multicast address and encryption key at any time, to effectively shut off a client's transmissions to every channel except the single channel for a while. In other words, once an individual multicast address and encryption key is defined for the single channel, a client 4 is able to switch channels without communicating with the server 22. As a result, a receiver may be required to listen to two multicast addresses (using two corresponding encryption keys) for each sender, where one of the pairs of addresses and keys is a normal broadcast address & key assigned to that sender and the other is an individual channel address & key. For efficiency, an individual address & key is not assigned until it is requested by a client who is sending A/V data over a single channel. Specifically, to support push-to-talk mode of microphone widgets 63, sources 85 request a multicast address and an encryption key for sending only to a particular channel using the get_transmitter_address() method in Table 1. Once the get_transmitter_address() method call returns an assigned address and key, a client is able to switch in and out of push-to-talk mode without any further communication with the server, thereby minimizing client-server latency. Should the address or key change due to a change in channel membership, the key manager sends the use_transmitter_address() notification in Table 2.

To minimize source transmission of A/V data, the key manager 25 keeps track of whether receivers 95, a sink's equivalent of a transmitter 94, currently exist for each device of each source 85. When an individual address & key for a channel is to be assigned and a user object has a device with a broadcast set containing only one transmitter on that channel, the broadcast address & key can be the same as the individual channel address & key. As soon as some other transmitter is added to the broadcast set, a new broadcast address & key is distributed which does not match the individual address & key. The key manager 25 sends the broadcast_has_receivers() notification whenever a set of receivers for a device's broadcast set starts or stops being empty. Similarly, individual transmitters are notified in a similar way with the transmitters_have_receivers() notification. For efficiency reasons, the key manager 25 uses the same multicast address for two transmissions if a group of receivers is identical. This requirement is the same for the key manager when it picks and distributes encryption keys. Thus, any given multicast address has only one encryption key in active use at a time.

In addition to assigning addresses and keys for sending out-of-band data, the key manager 25 provides a mechanism for sending small amounts of in-band data. In-band data is intended primarily as a means for a source 85 to send the name of a sound, image or video sequence that it wishes recipients to retrieve and display or play. For example in-band data can be used to generate audible sounds when a user enters a room. Such in-band data is sent to either a device's broadcast set using the broadcast_value() method, or to a specific channel using the transmit_value() method.

Table 3 lists methods key manager 25 provides for sinks 86, and Table 4 lists notifications provided by key manager 25 to sinks 86. (Note: as with other method calls, a sink parameter is an implied parameter of each method call.) Each sink 86 declares its intent to receive a specific A/V data stream sent through a channel 96 by creating a receiver 95 and associating it with the specific A/V data stream, using the create_receiver() and aim_receiver() receptively. For example, each video pane widget 67 (shown in Figures 3 and 5) displayed on a user's display terminal has a single associated receiver. In contrast, each speaker widget 68 has a set of associated receivers. For example, the set of associated receivers for a speaker widget for a room object may include all of the receivers pointed to A/V data streams of user objects in the room.

Table 3

Key manager methods called by sink

create_receiver() => receiver
aim_receiver(receiver, source, device, channel)
unaim_receiver(receiver)
destroy_receiver(receiver)

Table 4

Key manager notifications sent to sinks

use_receiver_addresses(receivers, list of {address, key})
receivers_have_transmitters(receivers, yes_or_no)

Any source or sink that is a member of a channel can use the channel_membership() method call to discover the complete membership of a channel. This method call relies on the key manager and not the channel manager to prevent the latter from maliciously giving different answers to different sources and sinks. Members of a channel can also use the channel_transmitters() and channel_receivers() method calls to find out what signals are available through a given channel and which of those signals are being received and which sinks are receiving those signals.

Communication over a channel is secure as long as the key manager generates each key distributed to sources and sinks. A channel manager could alternatively pick an address and key instead of having the key manager generate them. A channel manager is provided with this functionality in order to allow sessions generated from outside the central server to be viewed using the four A/V widgets (e.g. microphone, camera, video pane, and speaker). In this case, the key manager can not make any guarantees about who knows or has the keys, so the channel is marked as insecure. The channel_is_secure() method enables sources and sinks to determine whether a key, which has been distributed by the key manager, is secure.

2. Key Manager Data Structures

The key manager 25 is conditioned to notify changes of addresses and keys only when necessary. To do this the key manager must maintain several mappings that hold current states of channels, transmitters, devices, and receivers that are listed in Table 7. The mappings listed in Table 7 balance between redundant information and efficiency. The four mappings provide sufficient information to implement the method calls and notifications in Tables 1-6. When certain transitions or changes occur in the values of the sets or data structures in Table 7, appropriate notifications are sent to a channel, transmitter, device, or receiver. These notifications are sent by the key manager when changes occur to values in the data structures that the key manager maintains. These changes are generally triggered from a call by a source, sink, or channel manager to the key manager to perform some action such as creating or destroying a transmitter or receiver, changing where a transmitter or receiver is aimed, or changing the membership of a channel.

Table 7

Key manager data mappings

C[channel] => <members, xmtrs, rcvrs, address, key, watchers>

T[source] => <device, channel, is_broadcaster>

D[source, device] => <has_rcvrs, address, key>

Z[source, device, channel] => <xmtrs, rcvrs>

3. Widget Interface

When a microphone or camera widget is created, the widget makes a call to create_transmitter() in Table 1. The resulting transmitter is saved as part of the widget's private state. Two methods available on microphone and camera widgets are: set_device(device) and set_channel(channel). These two methods are used by applications to set from which of a user's local devices to get a signal and to which channel to send a signal, respectively. Microphone and camera widgets then make calls to set_transmitter_channel() and set_transmitter_device() as appropriate. It is expected that a user's client program will send back an event that the specific widget has been turned off when it receives a set_device() method call. This communication ensures that an application program running on the server will not cause a user to start sending from a different device without an explicit action from that user.

Server applications set which sources a speaker widget should listen to by calling the speaker widget method: set_sources(list_of {source, device, channel, volume}). When the set_sources() method is called, a widget creates a set of new receivers, one aimed at each of the given sources in the list. The volume arguments are passed onto the client program. The volume argument allows an application to specify the relative level of incoming audio, so that certain streams can be played more softly than others. This is useful, for example, when trying to play audio from sources that are virtually "farther away," such as in a nearby room. One difference between speaker widgets and video pane widgets is that, a single receiver for a video pane can be created and saved once the widget is created since a video pane widget can only receive at most one signal at a time. Thus, the video pane widget re-aims a receiver whenever the application changes the source to which the widget is listening to. To make this change at a client, applications call the set_source(source, device, channel) video pane method.

E. Illustrative Implementation

Figure 7 is a representation 140 of the communicator window 48 shown in Figure 3. As in Figure 3, the communicator window 140 represents a virtual room in which user "Ron" 142 and user "Pavel" 144 are communicating in a

receiver corresponds to a widget on each user's respective display terminal: Pavel's receiver 201 and Ron's receiver 207, which are respectively directed at the video pane widget 152, point to device 173 of source 170 through channel 180; Pavel's receiver 202 and Ron's receiver 208, which are respectively directed at the video pane widget 153, point to device 176 of source 171 through channel 180; Pavel's receiver 203 and Ron's receiver 209, which are respectively directed at the primary speaker widget 155, point to device 174 of source 170; and Pavel's receiver 204 and Ron's receiver 210, which are respectively directed at the primary speaker widget 155, point to device 177 of source 171.

The whisper or private channel 181 is created for push-to-talk microphone widget 146 on Ron's client workstation 4 and microphone widget 147 on Pavel's client workstation 4. Figure 8 shows the case of Pavel and Ron whispering to each other using channel 181. As noted above, channel 181 has a static membership list and therefore does not have a channel manager associated with it. To set up whisper channel 181, sources 170 and 171 create transmitters 186 and 189 respectively. In addition, sinks 178 and 179 create receivers 205-206 and 211-212 respectively. Transmitters 186 and 189, and receivers 205-206 and 211-212 are aimed at channel 181. Receivers 205 and 211 are set to point to device 174 of source 170, and receivers 206 and 212 are set to point to device 177 of source 171. Each client controls through explicit actions by a user's selection of a widget where to transmit A/V data. When a microphone widget is used as a push-to-talk button for the first time, the widget calls the key manager method get_transmitter_address() to have a multicast address and encryption key assigned for a specific transmitter channel. It will be appreciated that further channels 182, 183 may also be provided.

It will no doubt be appreciated that there are a number of possible manners in which to implement the key manager that could be used effectively with this media coordination system. What is required by this invention is a plurality of client workstations connected to a central server through a network. The central server coordinates streams of audio and video data between clients who multicast their A/V data over the network. The media coordination system combines automatic encryption, dynamic interconnection of streams of data, and user interface elements that provide clients with control over the ultimate destination of their A/V data. Even though a central server is coordinating where A/V information is being broadcast, each client workstation ultimately controls its broadcast.

The disclosed media coordination system may be readily implemented in software using object oriented software development environments that provide portable source code that can be used on a variety of hardware platforms. Alternatively, the disclosed system may be implemented partially or fully in hardware using standard logic circuits. Whether software or hardware is used to implement the system varies depending on the speed and efficiency requirements of the system and also the particular function and the particular software or hardware systems and the particular microprocessor or microcomputer systems being utilized. The system, however, can be readily developed by those skilled in the applicable arts without undue experimentation from the functional description provided herein together with a general knowledge of the computer arts.

The invention has been described with reference to a particular embodiment. Modifications and alterations will occur to others upon reading and understanding this specification taken together with the drawings. The embodiments are but examples, and various alternatives, modifications, variations or improvements may be made by those skilled in the art from this teaching which are intended to be encompassed by the following claims.

Claims

1. A method for dynamically controlling multiple channels of data in a multi-user collaborative system (2) having a central server (22) connected to a plurality of client workstations (4) over a network (8), comprising the steps of:

   displaying at each client workstation (4) a view on a room object (37) stored in an object database on the central server (22), the room object (37) being associated with a first channel stored in the object database;
   providing, at each client workstation (4), visual identification of each user object (36) located in a virtual room, each pair of user objects (36) located in the virtual room having associated therewith a whisper channel (102; 181);
   initiating, at a first client workstation (4), broadcast of data to each user object (36) located in the virtual room by selecting a first interface element displayed at the first client workstation (4), the first interface element being associated with the room object (37) and directing data to the first channel; and
   interrupting, at the first client workstation (4), broadcast of data transmitted over the first channel by selecting a second interface element displayed at the first client workstation (4), the second interface element being associated with a user object (36) at a second client workstation (4), said interrupting step initiating broadcast of data at the first client workstation (4) to the whisper channel (102; 181) associated with the user object (36) at the second client workstation (4).

2. A method according to claim 1, further comprising the step of terminating said interrupting step to resume broadcast

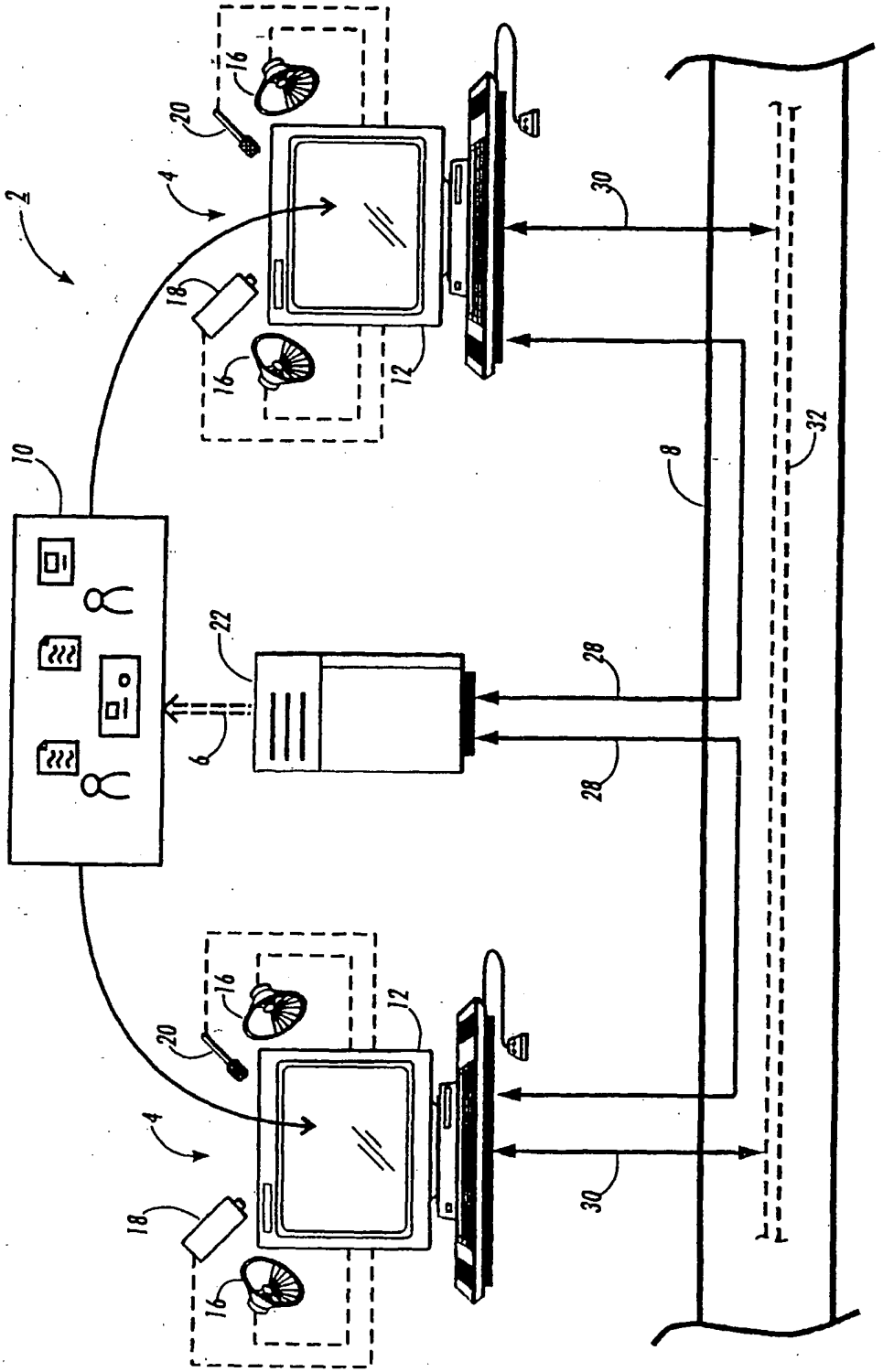

FIG. 1

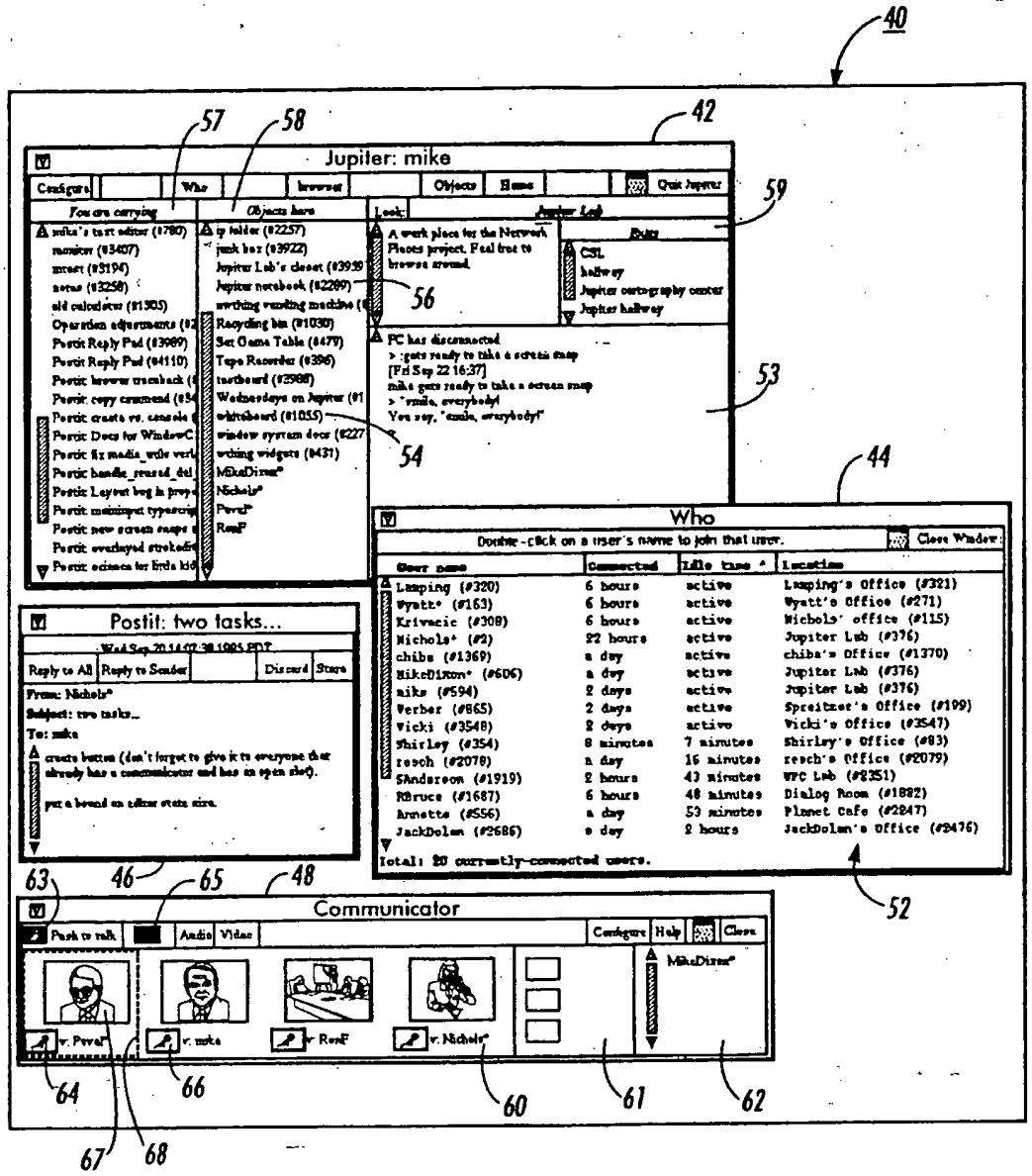
40
42
57
58
59
56
53
54
44
52
46
48
63
65
64
66
67
68
60
61
62
Jupiter: mike
Configure
Who
Browser
Objects
Home
Quit Jupiter
You are carrying
Objects here
Look:
Jupiter Lab
Rooms
CSL
hallway
Jupiter cartography center
Jupiter hallway
PC has disconnected
> :gets ready to take a screen snap
[Fri Sep 22 16:37]
mike gets ready to take a screen snap
> "smile, everybody!
You say, "smile, everybody!"
Who
Double-click on a user's name to join that user.
Close Window
User name
Connected
Idle time
Location
Lamping (#320) 6 hours active Lamping's Office (#321)
Wyatt* (#163) 6 hours active Wyatt's Office (#271)
Krivacic (#308) 6 hours active Nichols' office (#115)
Nichols* (#2) 22 hours active Jupiter Lab (#376)
chiba (#1369) a day active chiba's Office (#1370)
MikeDixon* (#606) a day active Jupiter Lab (#376)
mike (#594) 2 days active Jupiter Lab (#376)
Verber (#865) 2 days active Spreitzer's Office (#199)
Vicki (#3548) 2 days active Vicki's Office (#3547)
Shirley (#354) 8 minutes 7 minutes Shirley's Office (#83)
resch (#2078) a day 16 minutes resch's Office (#2079)
SAnderson (#1919) 2 hours 43 minutes WRC Lab (#2351)
RBruce (#1687) 6 hours 48 minutes Dialog Room (#1882)
Arnette (#556) a day 53 minutes Planet Cafe (#2847)
JackDolan (#2686) a day 2 hours JackDolan's Office (#2476)
Total: 20 currently-connected users.
Postit: two tasks...
Reply to All
Reply to Sender
Discard
Store
From: Nichols*
Subject: two tasks...
To: mike
Communicator
Push to talk
Audio
Video
Configure
Help
Close
MikeDixon*

FIG. 3

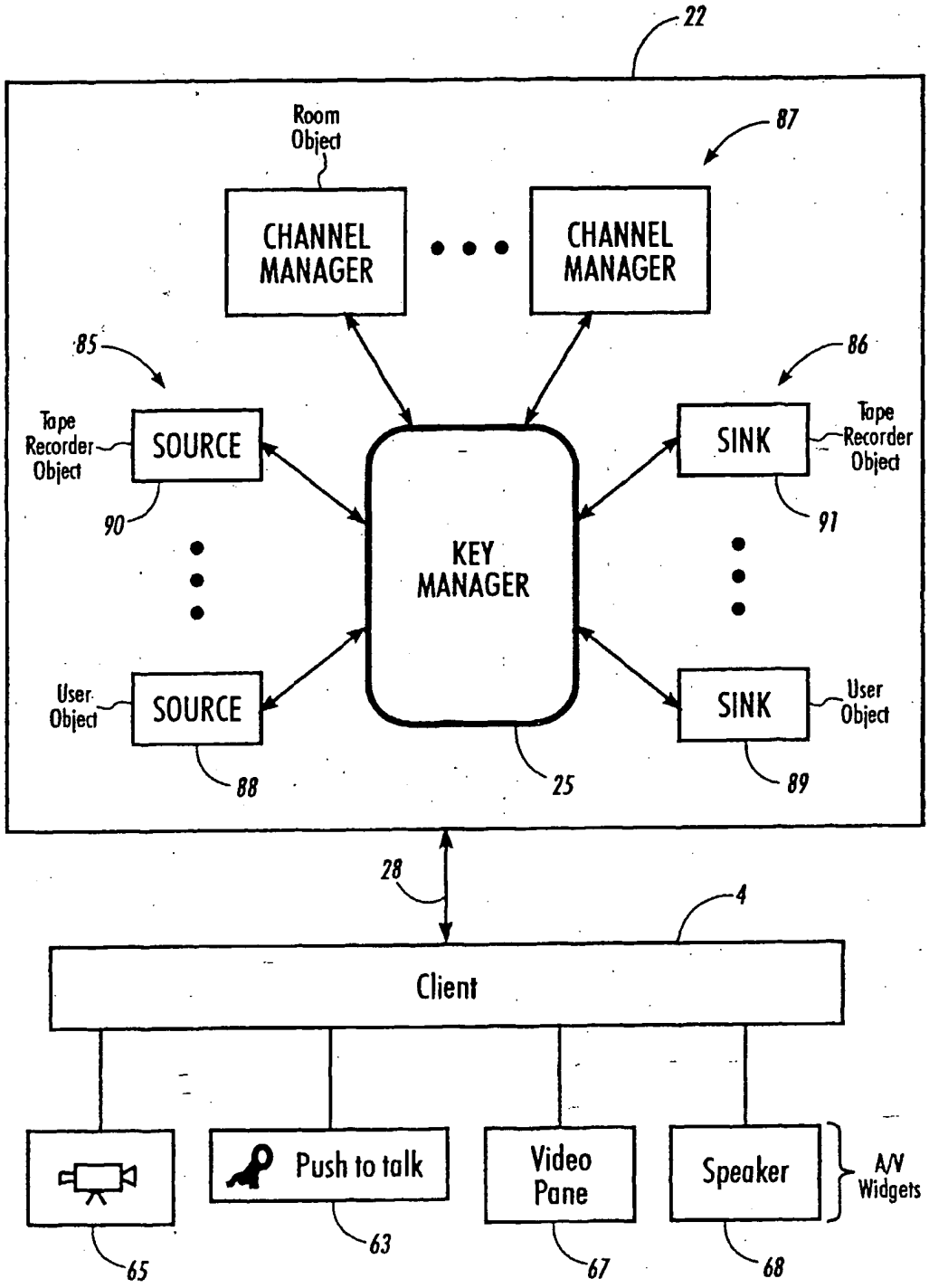
22
Room Object
87
CHANNEL MANAGER
CHANNEL MANAGER
85
86
Tape Recorder Object
SOURCE
SINK
Tape Recorder Object
90
91
KEY MANAGER
User Object
SOURCE
SINK
User Object
88
25
89
28
4
Client
Push to talk
Video Pane
Speaker
A/V Widgets
65
63
67
68

FIG. 5

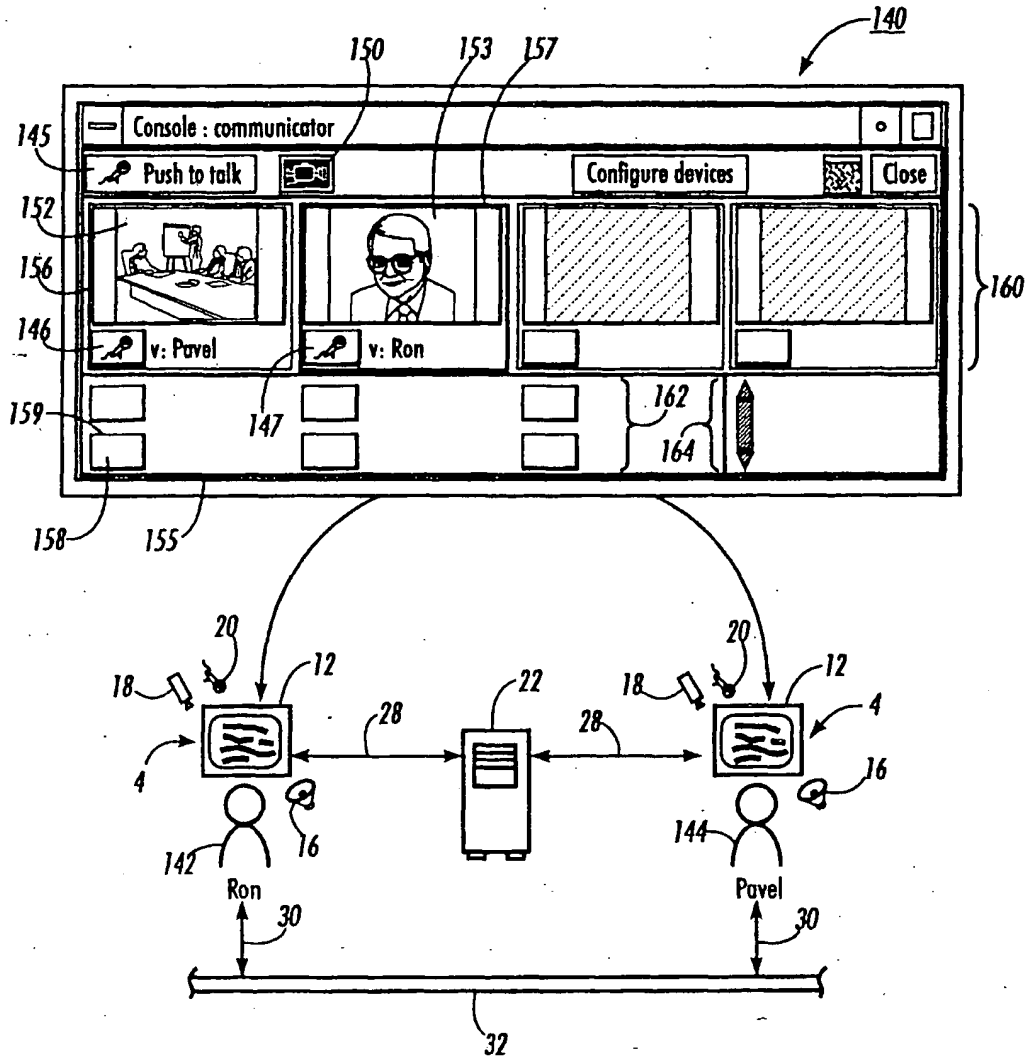
140
150
153
157
145
Console : communicator
Push to talk
Configure devices
Close
152
156
146
160
v: Pavel
v: Ron
159
147
162
164
158
155
20
18
12
22
28
4
16
142
144
Ron
Pavel
30
32

FIG. 7

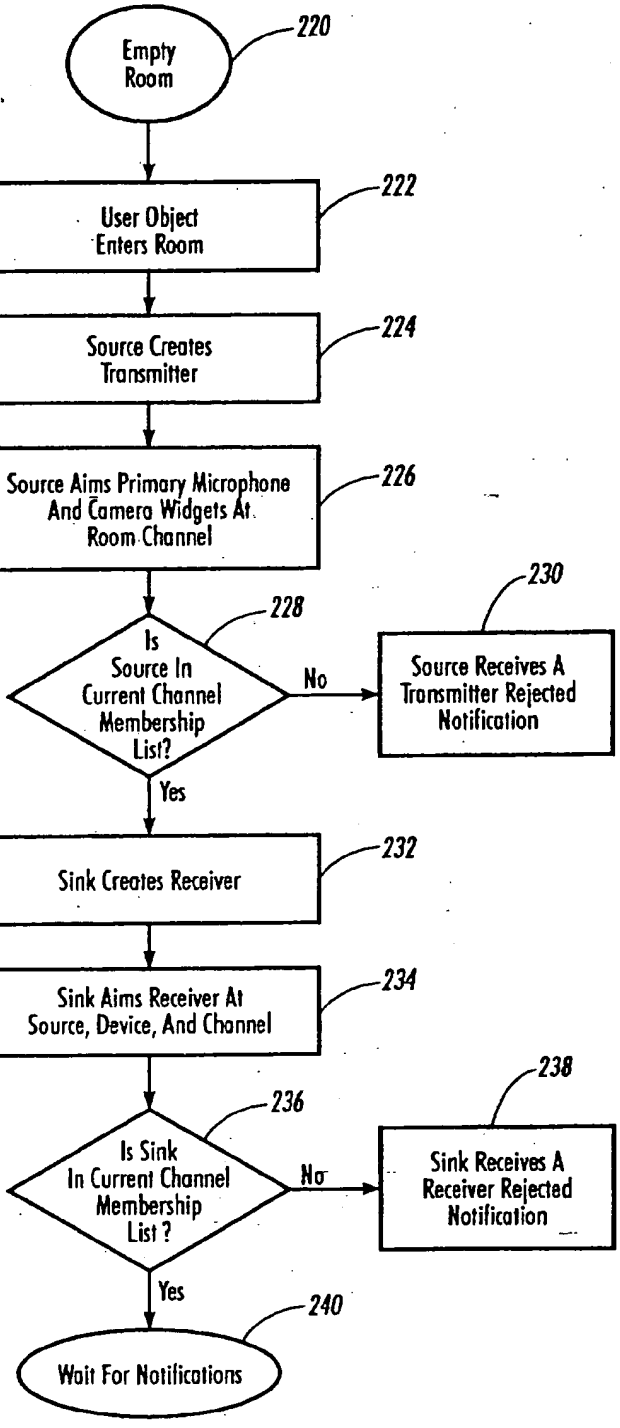
Empty Room
220
User Object Enters Room
222
Source Creates Transmitter
224
Source Aims Primary Microphone And Camera Widgets At Room Channel
226
Is Source In Current Channel Membership List?
228
No
Source Receives A Transmitter Rejected Notification
230
Yes
Sink Creates Receiver
232
Sink Aims Receiver At Source, Device, And Channel
234
Is Sink In Current Channel Membership List ?
236
No
Sink Receives A Receiver Rejected Notification
238
Yes
Wait For Notifications
240

FIG. 9